**Mini Project Report on**



**YOGA POSE DETECTION SYSTEM**



**Submitted in partial fulfillment of the requirement for the award of the degree of**

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

**Submitted by:**

**Student Name**  **University Roll No.**

**Tanmay Pandey 2021498**

***Under the Mentorship of***

**Ms Meenakshi Maindola**

**PROFESSOR**



**Department of Computer Science and Engineering**

**Graphic Era (Deemed to be University)**

**Dehradun, Uttarakhand**

**January-2024**



**CANDIDATE’S DECLARATION**

I hereby certify that the work which is being presented in the project report entitled **“Yoga Pose Detection System”** in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Computer Science and Engineeringof the Graphic Era (Deemed to be University), Dehradun shall be carried out by the under the mentorship of **Ms. Meenakshi Maindola, Professor**, Department of Computer Science and Engineering, Graphic Era (Deemed to be University), Dehradun.

Name : Tanmay Pandey University Roll no1 : 2021498

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Chapter No.** | **Description** | **Page No.** |
| Chapter 1 | Introduction | **1-3** |
| Chapter 2 | Literature Survey | **4-5** |
| Chapter 3 | Methodology | **6-11** |
| Chapter 4 | Result and Discussion | **12-16** |
| Chapter 5 | Conclusion and Future Work | **17** |
|  | References | **18** |

**Abstract**

The yoga pose recognition system uses computer vision, machine learning, and web development to instantly recognize yoga poses. MediaPipe helps store important content in CSV format. Logistic regression, random forests, and gradient boosting are all perfectly trained and tested with Standard Scaler. This helps develop algorithms that can be used for yoga fly detection using MediaPipe and OpenCV. Streamlit has an easy-to-use user interface that enables the front end; Python forms the basis of the backend. This little project provides a useful tool for yoga enthusiasts to learn about various combinations.

Keywords: Yoga pose detection, MediaPipe, computer vision, machine learning, logistic regression, random forest, gradient boosting, standard scaler, OpenCV, Streamlit, real-time recognition.

**Chapter 1**

**Introduction**

1. **Introduction**

Yoga is an ancient tradition that involves much more than physical exercise. It offers solutions to mental and physical health. But when life flies by, it can be easy to squeeze yoga into our schedule. This makes us a great person.

The wide world of technology sometimes hides its ability to improve our own health. Let's introduce the "Yoga Posture Detection System". The project was born from the idea that technology could make practicing yoga easier and more useful.

1. **Aim of the Project**

The goal is simple but beautiful: Understand and master yoga quickly. Yoga is more than stretching and bending; It's about finding balance and peace within ourselves. The program strives to make yoga more interactive and personal by providing instant feedback on form and body, which is essential for a meaningful yoga practice.

1. **Need of this Project**

Yoga is more than physical therapy; This is a good habit that can improve mental health and well-being. We tried a lot of things before deciding on the yoga set you see today. We started investigating the communication neural network (CNN) model and even combined it with the VGG16 architecture. While these techniques have their benefits, we find they fall short in addressing the nuances of different yoga poses.

1. **Relatively better method**

Enter our current method, which involves using MediaPipe Holistics in conjunction with machine learning algorithms to extract important content. Unlike traditional CNN architectures, this approach excels in capturing detailed nuances of yoga poses. Yoga involves subtle changes and by understanding the basic concepts we can create a more detailed and precise description of the poses. It's like having a virtual yoga instructor who provides personal guidance.

1. **Why better than CNN or pre-trained CNN**

CNN Rules have their place, but yoga poses often require more understanding. The combination of feature extraction and machine learning allows for more detailed analysis, including the relationship between different parts of the body. When we add new yoga poses, most of the time we have scalability issues or long model training time when the hardware is not very good. With this approach, our program takes less time to inculcate new poses and train models according to them. Mediapipe offers lots of solutions for other implementations also.

1. **Why Streamlit was used?**

Since not everyone is a web development expert, we turned to Streamlit to build the front end. Streamlit simplifies the process and allows us to create user-friendly interfaces without extensive web development knowledge. This ensures that our Yoga Sense can be used by many users, regardless of their skill level. It is a Python library used to create a frontend without much problem.

1. **Standardize using Standard Scaler**

It is important to place the Standard Scaler in our machine learning pipeline when searching for features. StandardScaler ensures that every feature on which our model is based conforms to the standard scale. This is important because it helps avoid some of the control over others due to difference in scale when using algorithms such as logistic regression, random forests, and gradient boosting. It helps to improve the stability of the model and the integration of the model, ultimately helping to improve the performance of yoga model.

**1.8.1 Logistic Regression:**

Think of logistic regression as a wise yoga guru evaluating each pose. This algorithm is like a teacher evaluating whether a student has built the model or not. It calculates the probability that a sample belongs to a particular class to ensure that the classification is correct. Logistic regression, like a wise yoga teacher, carefully pinpoints the details and makes it an important part of our project.

**1.8.2 Random Forest:**

Now imagine a group of different yoga teachers as a “random forest”. Each teacher (tree) in the forest has a say in the pose and they decide together. This teamwork ensures a positive and authentic experience. Just as different yoga teachers share their insights to deepen our understanding of the poses, each tree has its own perspective that leads to comprehensive consideration.

**1.8.3 Gradient Boosting:**

Finally, let us know about “Gradient Boosting”, a method that builds on the quality of the previous method. It's like yoga students learning from past mistakes and improving with each try. Gradient boosting combines multiple weak patterns to create a strong pattern that makes it easier to understand the cause over time. Just like a professional yoga teacher improves their skills, gradient boosting improves its model by learning from previous tests.

**1.9 Confusion Matrix:**

In our yoga pose detection system, the "Confusion Matrix " is our guide, providing a snapshot of the body's ability to recognize poses. By distinguishing between correct predictions (true positives and negatives) and incorrect predictions (false positives and negatives), it gives us information that will increase accuracy. This eye program acts like our compass, helping us find the right path of movement.

A screenshot of a computer

Description automatically generated

**Fig 1.1 showing Confusion Matrix**

**Chapter 2**

**Literature Survey**

**2.1 Computer Vision in Yoga Detection**

The integration of computer vision technology has played an important role in realizing yoga detection. Previous studies such as Cao et al. (2017)[1] laid the foundation of our method by proving the effectiveness of prediction from the key points. Using the MediaPipe library, we aim to capture the unique elements of yoga poses to provide a solid foundation for our yoga classification.

**2.2 Machine Learning in Yoga Recognition**

The combination of computer vision and machine learning has been shown to increase the accuracy of yoga recognition. Although research has always been started based on the CNN model, it needs to be developed by taking inspiration from the studies of Toshev and Szegedy (2014) [2]. The use of machine learning algorithms such as logistic regression, random forests, and gradient boosting allows our program to recognize the challenges of various yoga poses while also accounting for subtle relationships between key points of body.

**2.3 Traditional CNN models and limitations**

The beginning of our research involved creating CNN models. Although it may be useful in some cases, as seen in the study of Krizhevsky et al. (2012) [3] state that this model shows poor performance when applied to different and functional yoga poses. Fixed architectures have limitations in capturing complex details that need to be fully realized and they also show very slow speed for training time.

**2.4 Key Point Approach**

Newell et al. (2016) [4] keypoint-based technique inspired our method, representing a considerable advancement in it. Our bodies are better able to perceive the nuances implicit in yoga positions once the basics are understood. The use of MediaPipe brings more flexibility and accuracy to our yoga lighting solution and allows for the elimination of highlights as soon as possible.

**2.5 Standard scalers in machine learning pipelines**

When machine learning algorithms help us generate recognition correctly, standard scalers in our pipelines will not be ignored. Previous studies such as that of Pedregosa et al. (2011) [5] highlight the importance of feature scaling for algorithm stability. In our yoga recognition, the model scaler ensures that each feature is within a standard range, preventing errors and contributing to the robustness of our model.

**Chapter 3**

**Methodology**

**3.1 Tools Used:**

Frontend: Python ( Streamlit ), Microsoft Edge

Backend: Python ( Mediapipe , OpenCV , matplotlib , etc.)

Text Editor: VSCode

**3.2 Data Creation, Collection and Pre-Processing:**

We are going to use the MediaPipe Holistics Model which can be used to capture landmark features of hands, body, and face mesh. But in this model, we are only using body landmarks. Mediapipe holistic model is used to capture 33 Different landmark points.

A drawing of a person with red dots

Description automatically generated

**[6]Fig 3.2.0 Showing Key Landmarks of MediaPipe Pose**

**3.2.1 Creation of Database**

When createDatabase.py is run, it checks whether the coordinates.csv file exists or not; if the file exists, then it prints that the database already exists, otherwise, it captures the landmarks from the live camera feed and then creates a list which is then appended at the beginning with column name as the class name: which will denote name of the yoga pose.

**A screenshot of a computer

Description automatically generated**

**Fig 3.2.1.1: showing an appended list**

After that, the top row is added to the coordinates.csv file, and the database is created.

A diagram of a data flow

Description automatically generated

**Fig 3.2.1.2 showing a flowchart of checkAndCreate.py**

**3.2.2 Adding Yoga Poses to DataBase**

When add.py is run it first opens the coordinates.csv database and inputs the yoga pose name by the user. Then it uses OpenCV to record the live camera feed. It uses the MediaPipe Holistics model to capture body landmarks which are then modified according to the coordinates.csv file in a list which is later appended to it.

The camera is automatically closed after 10 seconds timer.

A diagram of a program

Description automatically generated

**Fig 3.2.2.1 showing flow-chart of add.py**

**3.2.3 Outlier Detection and Cleaning:**

This coordinates.csv is not required to be checked for any outliers because if any landmarks are not visible then their visibility will become zero and it is perfectly normal. In terms of data cleaning, there is no requirement as data is almost at the same distribution inside the data frame.

**3.3 Data Splitting and Model Training and Testing**

After running trainEvaluateAndSave.py we have our data coordinates.py which we are going to split our data into training data (70%) as well as testing data(30%) using scikit learn train test split and after that, we will save our training data as well as testing data inside train.csv and test.csv. Now we will create different functions to perform different tasks. We will create a pipeline where we are first going to use StandardScaler and then our other models which are Logistic Regression Classifier, Random Forest Classifier, and Gradient Boosting Classifier.

**A screenshot of a computer code

Description automatically generated**

**Fig 3.3.1 shows pipeline steps**

After we create the pipelining, we will train our models one by one and we will save different accuracy matrices. After training our model we are going to evaluate it and then from the given 3 models we are going to select the best model which has the highest accuracy. We will save the selected model as best\_model.pkl. We have used the pickle library because it has good compatibility with front-end web development.

After getting the best model we will print its pipeline and the name of the classifier which is being used in the best\_model.pkl

Performance Matrices used for the evaluation of the performance of each model are :

[7] Accuracy: (TP + TN) / (TP + TN + FP + FN) ………. (3.3)

[8] Precision: TP / (TP + FP) ………. (3.3)

[9] Recall: TP / (TP + FN) ………. (3.3)

[9] F1 Score: 2 \* (Precision \* Recall) / (Precision + Recall) ………. (3.3)

As our Dataset is not, unbalanced, we will currently use Accuracy for model selection.

If the dataset becomes unbalanced in the future, we can use the F1 score.

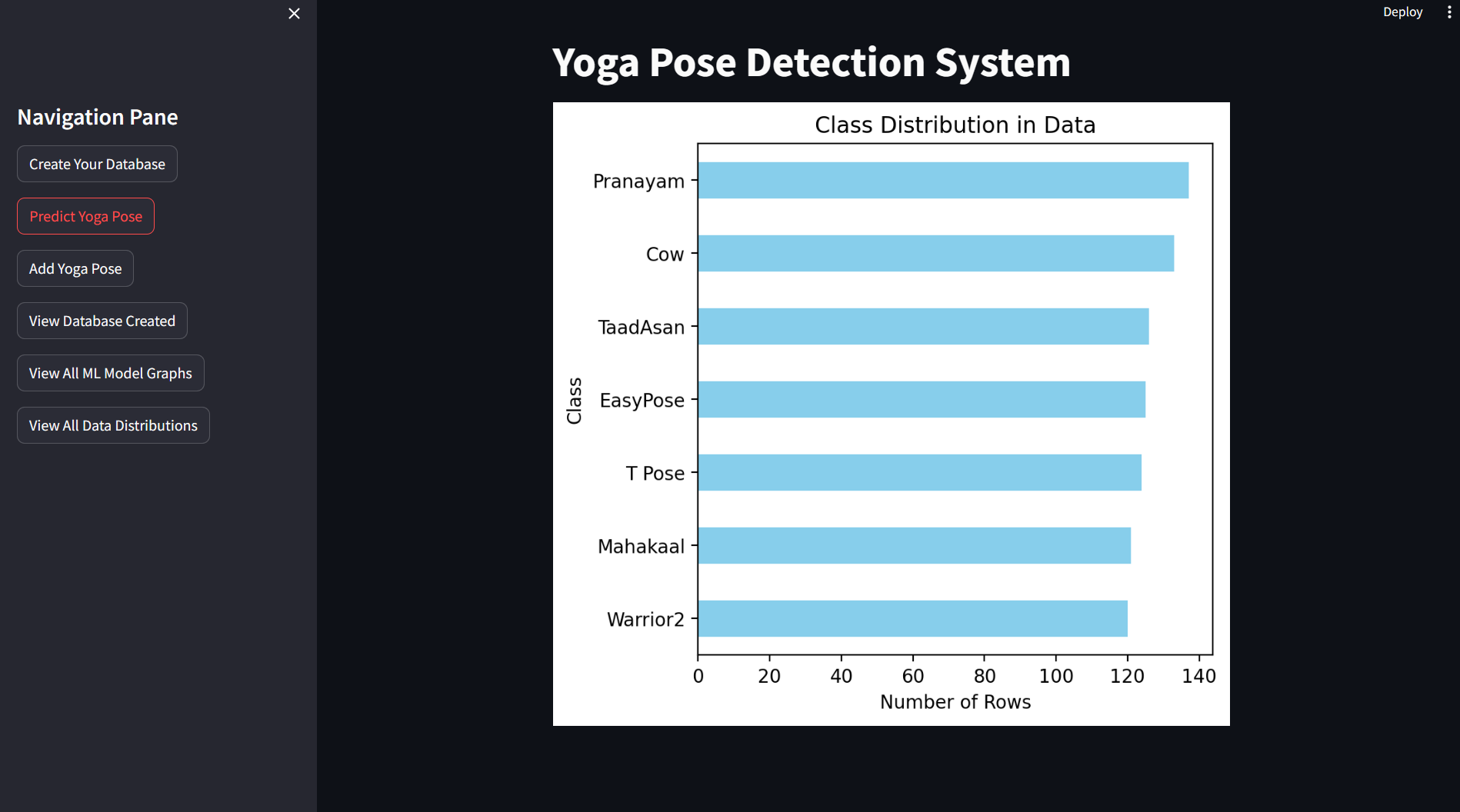
.

A screenshot of a computer flowchart

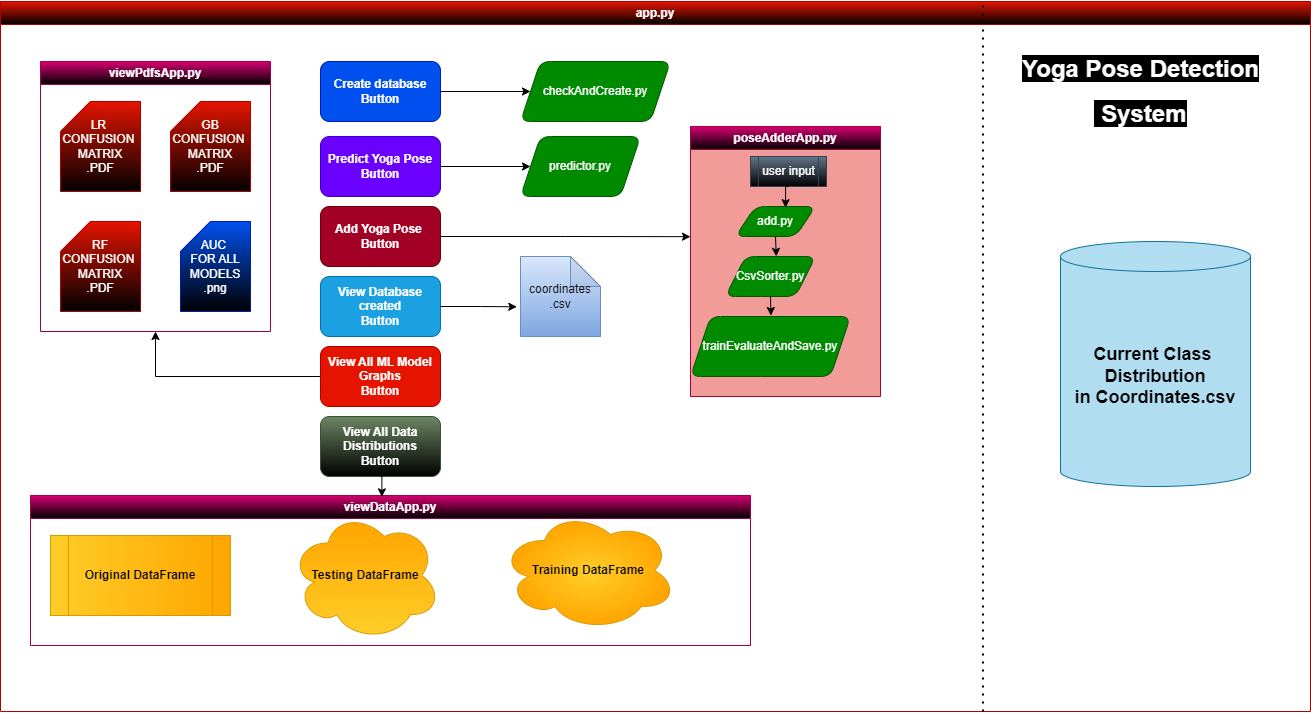
Description automatically generated

**Fig 3.3.2 shows the flowchart of trainEvaluateAndSave.py**

**3.4 Front-end using Streamlit**

****

**Fig 3.4.1 shows the front end created using Streamlit**

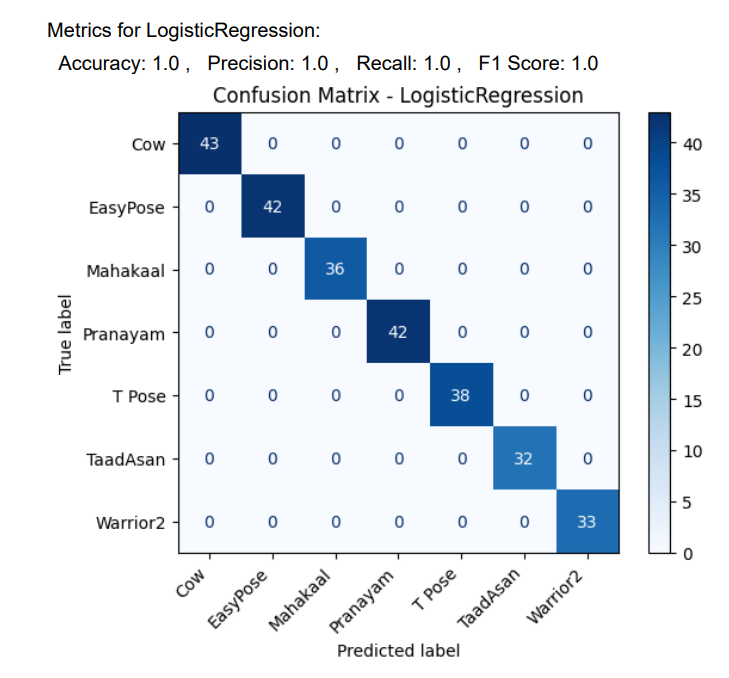
We have used the Streamlit library of Python which allows the creation of the front page , buttons, checkboxes, sliders, etc. We have used this because of its ease of access and handleability. We have created different buttons, some go to another webpage, some give predictions, and some give data.  
at the homepage, we will get the current data distribution in your current dataset. 

**Fig 3.4.2 shows the flowchart of the Streamlit application**

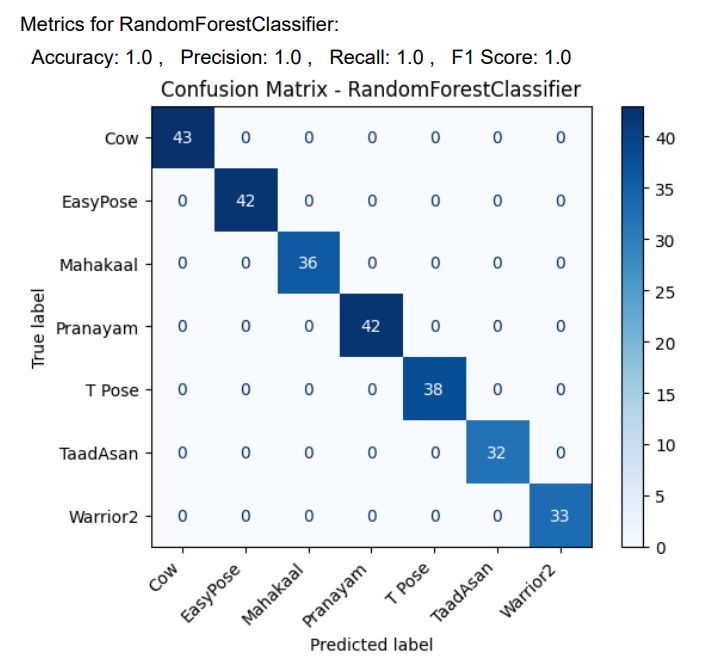
**Chapter 4**

**Result and Discussion**

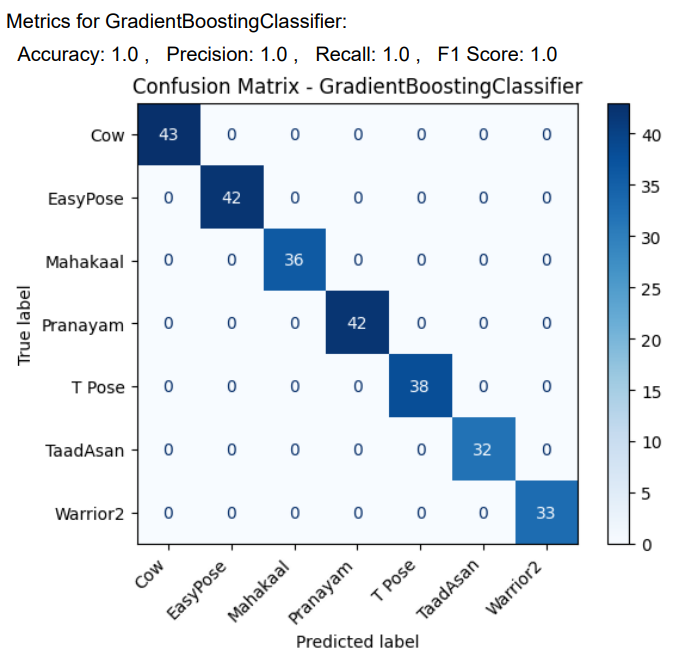
After our trainEvaluateAndSave.py file Has completed its execution it will show training ended. Now we can view our training data frame, our testing data frame, and our original data frame in .csv file. We have also saved the plot confusion matrix of all the model pipelines with their accuracy, F1 score, recall, and precision in a PDF format, which can be viewed later.

****

**Fig 4.1 shows the confusion matrix of the Logistic Regression Classifier**

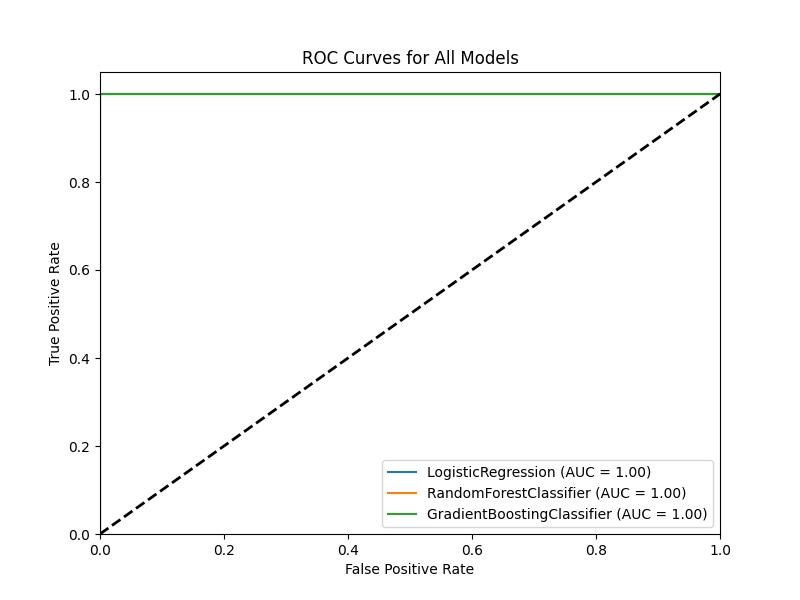
****

**Fig 4.2 shows the confusion matrix of the Random Forest Classifier**

****

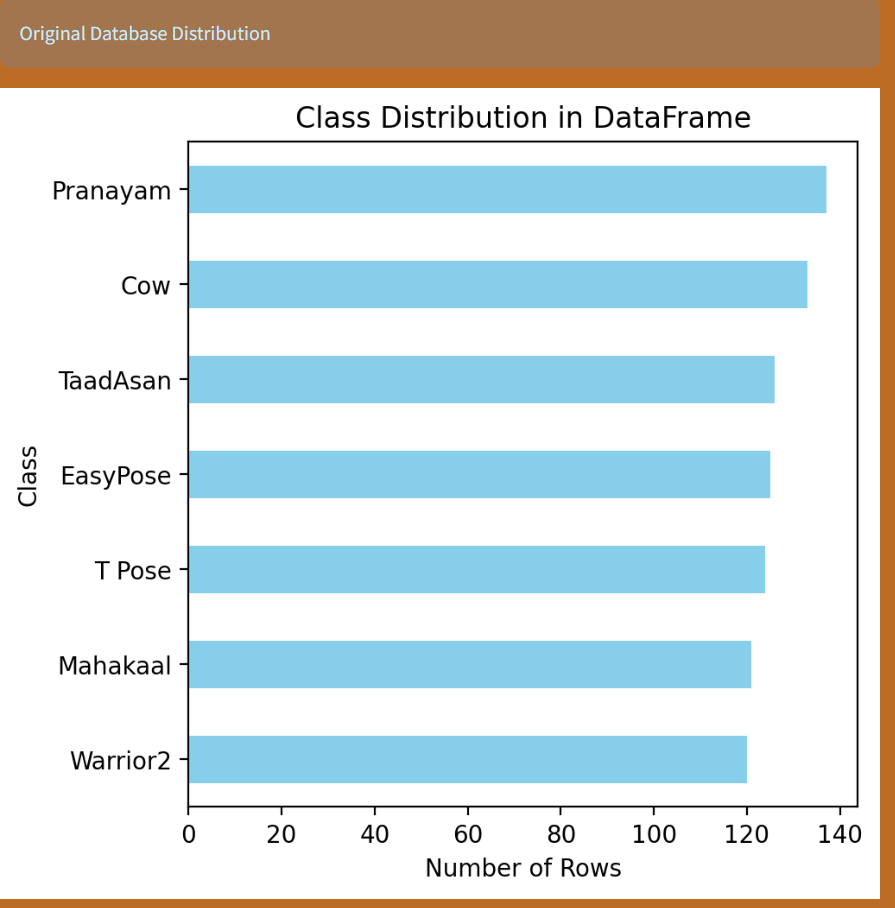
**Fig 4.3 shows the confusion matrix of the Gradient Boosting Classifier**

We have also saved the ROC vs AUC Curve plot as a comparison to check which model is performing best under an unbalanced data set if formed in the future(for scalability).

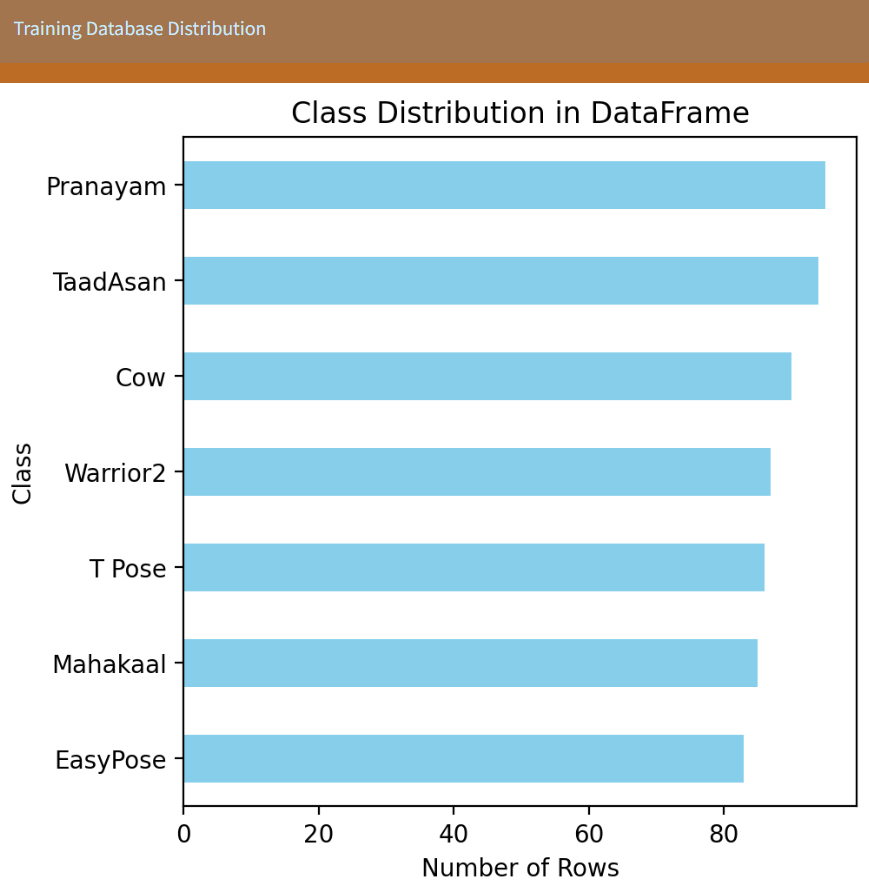


**Fig 4.4 shows the Area Under the Curve for all models**

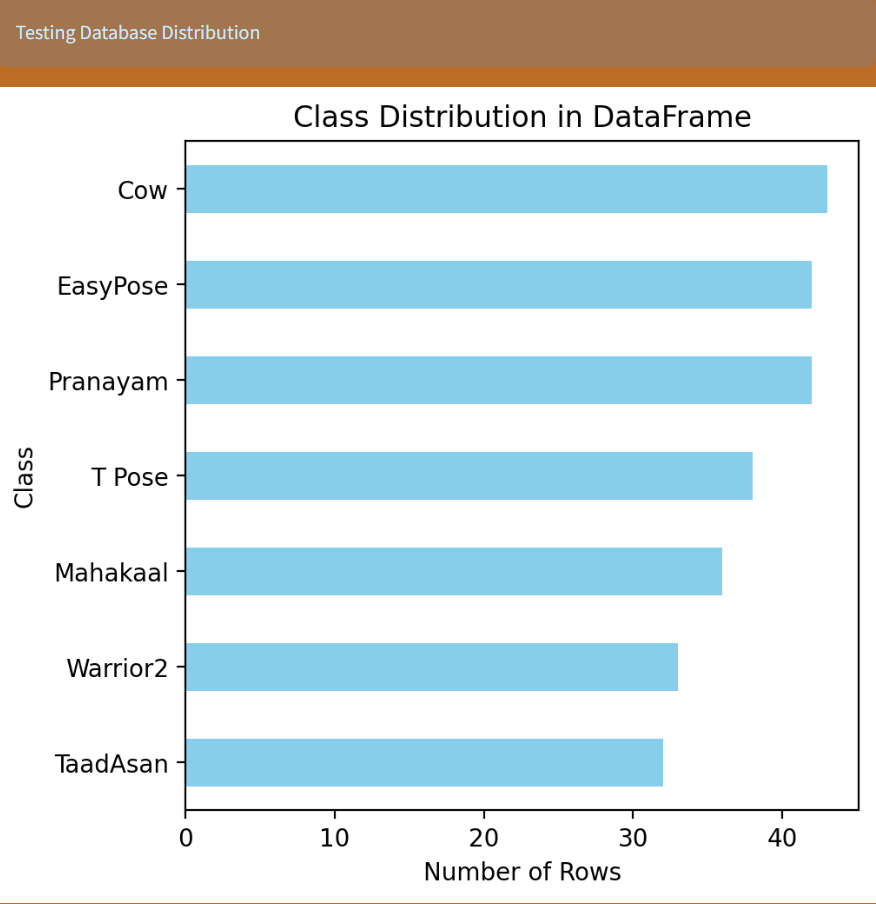
The visualization of our data distribution is done inside Streamlit app

****

**Fig 4.5 shows class distribution in the original data frame**

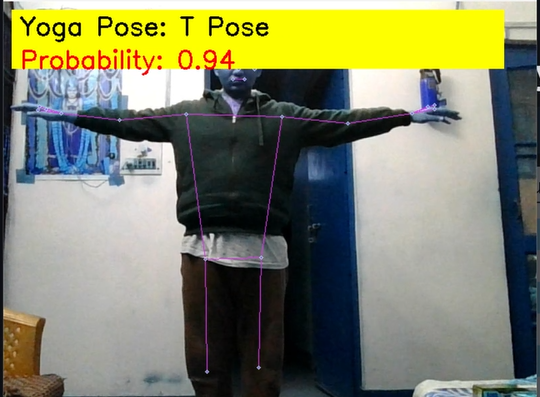
****

**Fig 4.6 shows class distribution in the Training data frame**

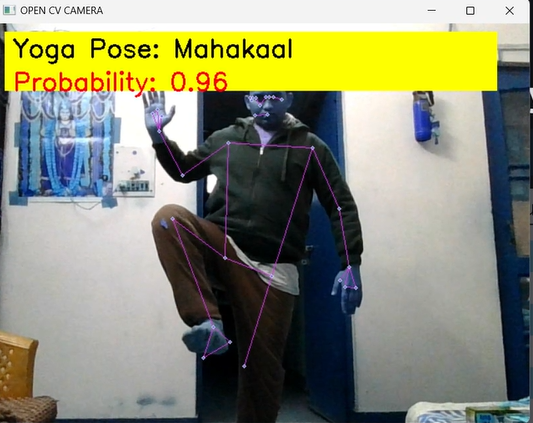
****

**Fig 4.7 shows class distribution in the Testing data frame**

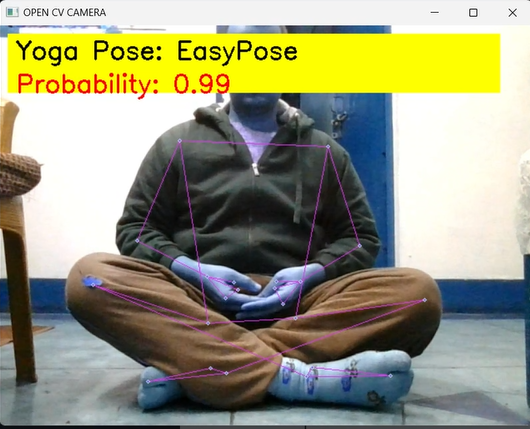
Now at last we have some real-time Yoga Pose detections.

****

**Fig 4.8.1 shows T-Pose Detection**

****

**Fig 4.8.2 shows MahaKaal Asan**

****

**Fig 4.8.3 shows Easy-Pose or Sukh Asan**

**Chapter 5**

**Conclusion and Future Work**

**5.1 Conclusion**

As we end our journey with the yoga guesser system, we have successfully combined ancient yoga skills with technology. By combining computer vision, machine learning, and web-based tools, we created a system that can instantly identify and measure yoga poses.

Our change to the traditional rules regarding the use of MediaPipe keys is important. It helps us overcome the difficult problems of yoga movement. Machine learning pairs like Logistic Regression and Random Forest and a little help from the Standard Scaler improve the accuracy of our system, especially on balanced data.

Making it easy for users to use is our top priority. Thanks to Streamlit, our interface is suitable for both tech-savvy and non-tech-savvy users. We don't just focus on technology; We focus on the same work. We want the system to be efficient and consider everyone's health.

**5.2 Future Work:**

* Expand the pose library: Increase our list of well-known poses covering a variety of yoga styles and traditions.
* Personalized Yoga Plans: Allow the system to learn from previous users and create personalized yoga plans.
* Multimodal feedback: Add audio and visual cues for better feedback on breathing and adaptation.
* Integration of user community: Create a place in the system where users can share information and support.
* Continuous model training: Make our machine learning models smarter by changing customer needs.
* Cross-device compatibility: Make sure our system works well on different devices such as phones, tablets and devices.

**References**

[1] Cao, Z., Simon, T., Wei, S. E., & Sheikh, Y. (2017). Realtime multi-person 2D pose estimation using part affinity fields.

[2] Toshev, A., & Szegedy, C. (2014). DeePose: Human Pose Estimation via Deep Neural Networks.

[3] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks.

[4] Newell, A., Yang, K., & Deng, J. (2016). Stacked Hourglass Networks for Human Pose Estimation.

[5] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine learning in Python.

[6] Google. (n.d.). Pose Landmarks Index. Retrieved from <https://developers.google.com/static/mediapipe/images/solutions/pose_landmarks_index.png> (Accessed on [18/1/2024])

[7] Christopher M. Bishop. Pattern Recognition and Machine Learning. Springer, 2006. (Section on Evaluation Metrics)

[8] Ian H. Witten and Eibe Frank. Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, 2005. (Chapter 6: Model Evaluation)

[9] Scikit-learn documentation: http://scikit-learn.org/stable/modules/model\_evaluation.html. (Metrics module documentation)